

USAISEC

US Army Information Systems Engineering Command
Fort Huachuca, AZ 85613-5300

1

AD-A270 034



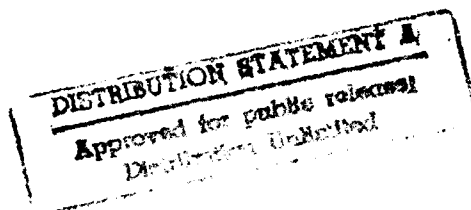
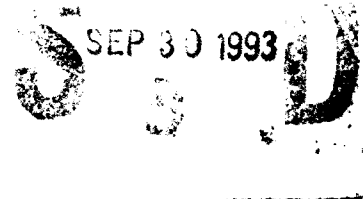
U.S. ARMY INSTITUTE FOR RESEARCH
IN MANAGEMENT INFORMATION,
COMMUNICATIONS, AND COMPUTER SCIENCES

AIR MICS

Integrated Office Information System (IOIS) Summary Report: Integration Strategy for Distributed Environment

ASQB-GM-90-025

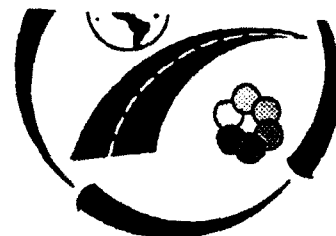
MAY 1990



93-22588



AIRMICS
115 O'Keefe Building
Georgia Institute of Technology
Atlanta, GA 30332-0800



REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS NONE		
2a. SECURITY CLASSIFICATION AUTHORITY N/A			3. DISTRIBUTION/AVAILABILITY OF REPORT N/A		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A					
4. PERFORMING ORGANIZATION REPORT NUMBER(S) ASQB-GM-90-025			5. MONITORING ORGANIZATION REPORT NUMBER(S) N/A		
6a. NAME OF PERFORMING ORGANIZATION AIRMICS		6b. OFFICE SYMBOL (If applicable) ASQB-GM		7a. NAME OF MONITORING ORGANIZATION N/A	
6c. ADDRESS (City, State, and Zip Code) 115 O'Keefe Bldg. Georgia Institute of Technology Atlanta, Ga 30332-0800				7b. ADDRESS (City, State, and ZIP Code) N/A	
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AIRMICS		8b. OFFICE SYMBOL (If applicable) ASQB-GM		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, State, and ZIP Code) 115 O'Keefe Bldg. Georgia Institute of Technology Atlanta, GA 30332-0800				10. SOURCE OF FUNDING NUMBERS	
				PROGRAM ELEMENT NO. 62783A	PROJECT NO. DY10
				TASK NO. 05	WORK UNIT ACCESSION NO.
11. TITLE (Include Security Classification) Integrated Office Information System (IOIS) Summary Report: Integration Strategy for Distributed Environment					
12. PERSONAL AUTHOR(S) Dr. Olivia R. Liu Sheng & Dr. Kuni Higa					
13a. TYPE OF REPORT		13b. TIME COVERED FROM _____ TO _____		14. DATE OF REPORT (Year, Month, Day) May 1990	
15. PAGE COUNT 40					
16. SUPPLEMENTARY NOTATION					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUBGROUP			
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
<p>The key to integrated office support in a distributed environment is the integration of heterogeneous databases used at different locations for various purposes. The core of heterogeneous database integration is at the logical level. This report presents an approach to logical integration of multiple databases.</p>					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT <input checked="" type="checkbox"/> UNCLASSIFIED/UNLIMITED <input type="checkbox"/> SAME AS RPT. <input type="checkbox"/> DTIC USERS			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED		
22a. NAME OF RESPONSIBLE INDIVIDUAL Michael Evans			22b. TELEPHONE (Include Area Code) 404/894-3107		22c. OFFICE SYMBOL ASQB-GM

This research was performed for the Army Institute for Research in Management Information, Communications and Computer Science (AIRMICS), the RDTE organization of the U.S. Army Information Systems Engineering Command (USAISEC). This research is not to be construed as an official Army position, unless so designated by other authorized documents. Material included herein is approved for public release, distribution unlimited. Not protected by copyright laws.

Accession For	
ADP	<input checked="checked" type="checkbox"/>
LES	<input type="checkbox"/>
U	<input type="checkbox"/>
Dist.	
A-1	

THIS REPORT HAS BEEN REVIEWED AND IS APPROVED

s/ James Gantt
James Gantt
Chief, MISD

s/ John R. Mitchell
John R. Mitchell
Director
AIRMICS

Integrated Office Information System (IOIS) Summary Report: Integration Strategy for Distributed Environment

Dr. Olivia R. Liu Sheng

Department of Management Information Systems

College of Business and Public Administration

University of Arizona

Tucson, Arizona 85721

602-621-2748

Dr. Kuni Higa

Georgia Institute of Technology

College of Management

Atlanta, GA 30332

404-894-4365

May 6, 1990

⁰†Submitted to the Army Institute of Research in Management Information, Communications and Computer Science (AIRMICS), Atlanta, GA. Grant #: DAKF-11-88-C-0021.

Executive Summary

The key to integrated office support in a distributed environment is the integration of heterogeneous databases used at different locations for various purposes. The core of heterogeneous database integration is at the logical level. This report presents an approach to logical integration of multiple databases.

Logical Integration of Multiple Databases

Kunihiko Higa
Georgia Institute of Technology
College of Management
Atlanta, GA 30332
Tel: (404) 894-4365
Bitnet: KHIGA@GTRI01

1. Introduction.

Existing DB schema integration methods minimally, if at all, discuss the role of data models in their integration processes. We believe that the use of an effective data model significantly enhance designers ability to identify and analyze integration problems. In this paper, the Structured Object Model (SOM) is used as the backbone data model throughout the logical DB schema integration process. SOM also represents the global schema when the integration process is completed. This paper describes SOM-based integration overview first. Then notion of SOM is introduced, followed by discussion of SOM-based DB schema integration phases. Examples of schema integration using SOM are described at the end.

2. The SOM-based Logical Schema Integration Overview.

The SOM-based logical schema integration consists of four basic phases as depicted in Figure 1. the first phase, preintegration, requires existing schemas, data dictionaries, other related documents, and users as inputs. The main objective of this phase is to represent existing schemas and their usage in uniform manner. The output of this phase is a set of SOM diagrams, each SOM diagram represents an existing schema. These diagrams are used at the next phase, schema comparison, to identify homonyms and synonyms of objects, attributes, and structures. This phase classifies objects and attributes into identicals/equivalents, weak conflicts, and strong conflicts. In the third phase, conflict resolution, both weak conflicts

and strong conflicts are resolved and standardized schemas, schema mapping/transformation, and exception handlings are defined. Then those standardized schemas are integrated at the last phase, integration.

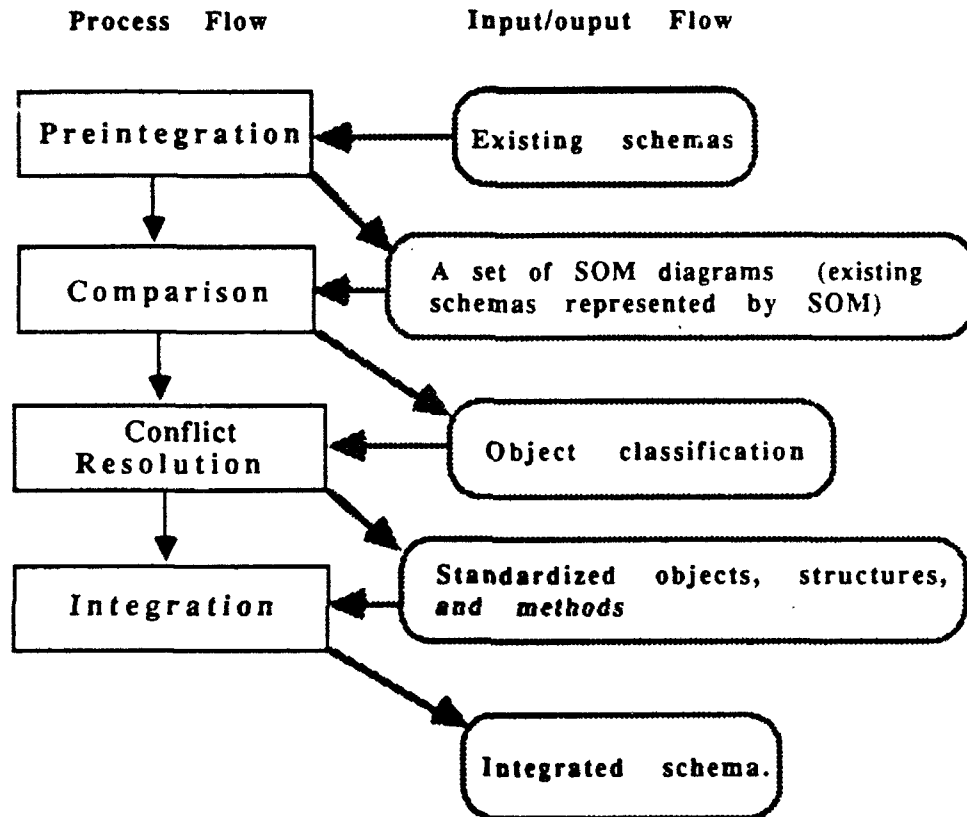


Figure 1. SOM-based Integration Overview.

In this paper, we assume that the logical schema integration process is performed on existing data bases. However, the methodology described in this paper is easily applicable to new data bases when each DB schema is developed using SOM-based DB design methodology [Higa and Liu Sheng, 1989].

3. Structured Object Model (SOM).

Structured Object Model (SOM), originating from System Entity Structure (SES) [Belogus, et al., 1980], has been developed as an analysis, design, and navigation tool for database applications [Higa, 1988, Chee, 1989, Higa and Liu Sheng, 1989, and Higa, 1990]. Throughout the integration process, SOM is used as a common representation scheme and thus the consistency between the global schema and individual schemas is greatly improved. In addition, because of its hierarchical structure and decomposition process, SOM naturally generates layered levels of abstraction in its integration process. Hence SOM facilitates accomplishment of the integration of multiple databases in a structured and top-down fashion.

In the following subsections, the basic components and constructs of SOM are briefly described.

3.1. Basic Components of SOM.

SOM represents data semantics using objects, attributes, and three types of relationships: aspect (property), specialization, and collection. An object set is a collection of events or objects about which users wish to collect and store information. In SOM, objects are represented by frames. However, there is a distinction between an object and an object set. Projects, employees, and equipments are examples of object sets. Attributes are used to describe objects by providing them with descriptive properties such as name, shape, color, etc. There are two basic types of attributes: identifiers and descriptors. An identifier uniquely identifies an object occurrence, and descriptors describe the state of the object occurrence. SOM also has an extended attribute type, called a method, which represents behavioral aspects of objects. For example, update operations and constraints can be encapsulated into an object as its extended attributes. Relationships represent associations among objects in the real world. Semantic meaning of relationships is indicated by the connectivity between objects (one-to-one, one-to-many, and many-

to-many). The graphical representations of the basic components in SOM are summarized in Figure 2.



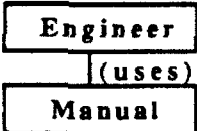
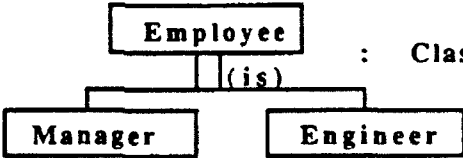
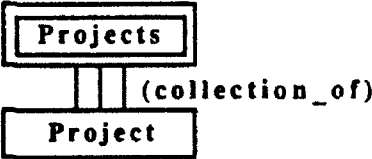
COMPONENT	REPRESENTATION
Object Set	 : Collection of similar objects.
Object	 : Single object
Attribute	
Identifier	● <u>Project ID</u> : Underlined attribute
Descriptor	● Project_Name
Method	
Constraint and Operation	*label: IF Cond. THEN Action
Membership	*OWNER(Name List), *MEMBER(Name List)
Relationship	
Aspect	 : Property relationship
Specialization	 : Classification
Collection	

Figure 2. Fundamental Components in SOM.

Relationships in SOM can be categorized as aspect relationships, specialization relationships, and collection relationships.

1. Aspect

Aspects describe owner-property or owner-member relationships. An object can be decomposed into the properties and members that belong to it. The term "aspect" has been adopted by reason of the fact that properties and members can be thought of as the aspects of an object. An aspect decomposition is represented in SOM diagrams by a single vertical link between the "parent object" and "child objects" and can be read as "has-a" in the top-down direction or as "is-part-of" in the bottom-up direction. The association name, such as "has" in Figure 3, may be attached the aspect link in order to clarify its semantics.

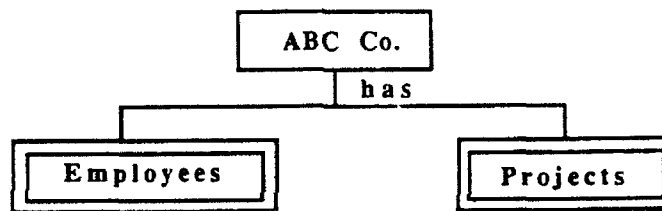


Figure 3. Aspect Relationships.

A relationship such as: "ABC Co. has employees and projects" can be expressed in the SOM diagram in Figure 3.

2. Specialization (Taxonomy)

Taxonomic knowledge represents the way in which objects are categorized into subclasses. In contrast to other decomposition structures, the taxonomic structure carries with it an inheritance principle: the specialization object inherits the substructure of its generalized superior (parent). Thus, what is true for "Employee" is also true for its special types "Manager" and "Engineer".

Consequently, a specialization object has at least the same number of attached variables and methods as its parent. However, the value of each inherited variable can be locally initialized. Otherwise, it will assume the default value from its parent. In Figure 4, all subclasses of "Employee" inherit attached variables of "Employee".

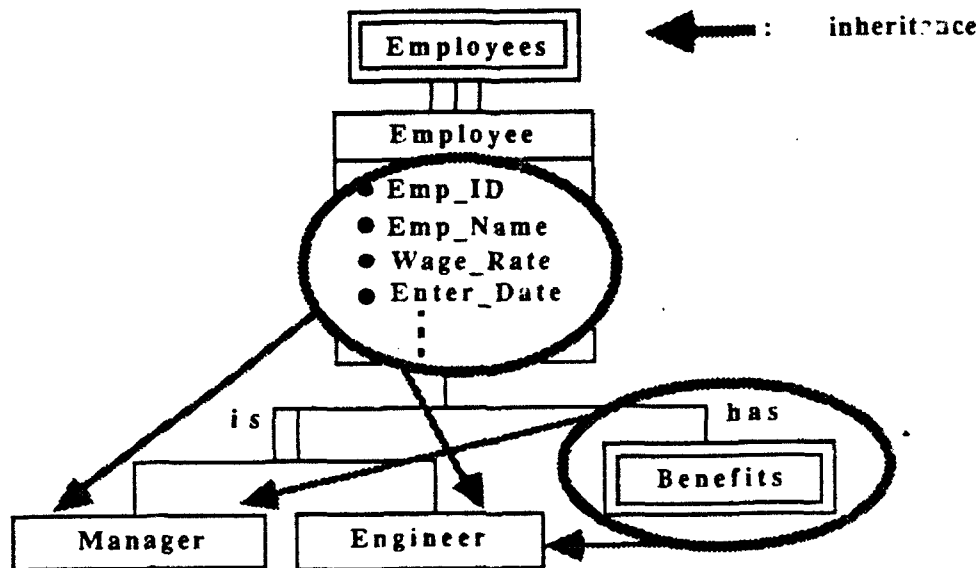


Figure 4. Specialization Relationships.

A specialization object may have additional properties not shared by its generic parent nor by its siblings. Indeed, it is the additional properties that distinguish one subclass from the others. A taxonomy (specialization) is denoted by double vertical lines between the parent and specialization objects in SOM diagrams and it should be read as "is-a-kind-of" in the bottom-up direction. For example, the fact that "Manager and engineer are kinds of employee" can be described in a SOM diagram as shown in Figure 4. The association name, such as "is" in Figure 4, may be attached the specialization link in order to clarify its semantics.

3. Collection

One feature of the Structured Object Model is the use of the object set (multiple object) concept. For example (see Figure 4), "Employees" is an object set, i.e., it is composed of one or more individual objects of the type "Employee". An object set has to be decomposed into a single object by collection relationship (represented by triple vertical lines in an SOM diagram) before it is further decomposed. SOM's collection relationship will help designers clarify the difference between a single object and an object set.

The type of connectivity (one-to-one, one-to-many, or many-to-many) between two objects, A and B, can be determined by checking both A-to-B and B-to-A relationships. For example, if an engineer uses many manuals and the same manual is used by many engineers, then the relationship between an object engineer and an object manual is many-to-many. Finally, a collection relationship decomposes an object set into individual objects. This decomposition is necessary to make the connectivity between objects clear.

3.2. Constructs of SOM

SOM is constructed by decomposition and coupling. The decomposition process generates the structure of an object, while coupling determines how objects are coupled. Stamp-coupling [Schneyer, 1984] is the coupling mechanism used in SOM. After an object has been defined in the structure, the mechanism couples later appearances of the object with its original definition. As a result, the mechanism ensures that an object will not be decomposed more than once.

The constructs of SOM are based on the degree and the connectivity of a relationship, and the membership class is determined according to the classification by Teorey, et al. [1986], as depicted in Figure 5.

1. Degree of a relationship is determined by the number of objects participating in a relationship. An n-ary relationship is of degree n (e.g., binary and ternary relationships are degrees of 2 and 3, respectively).

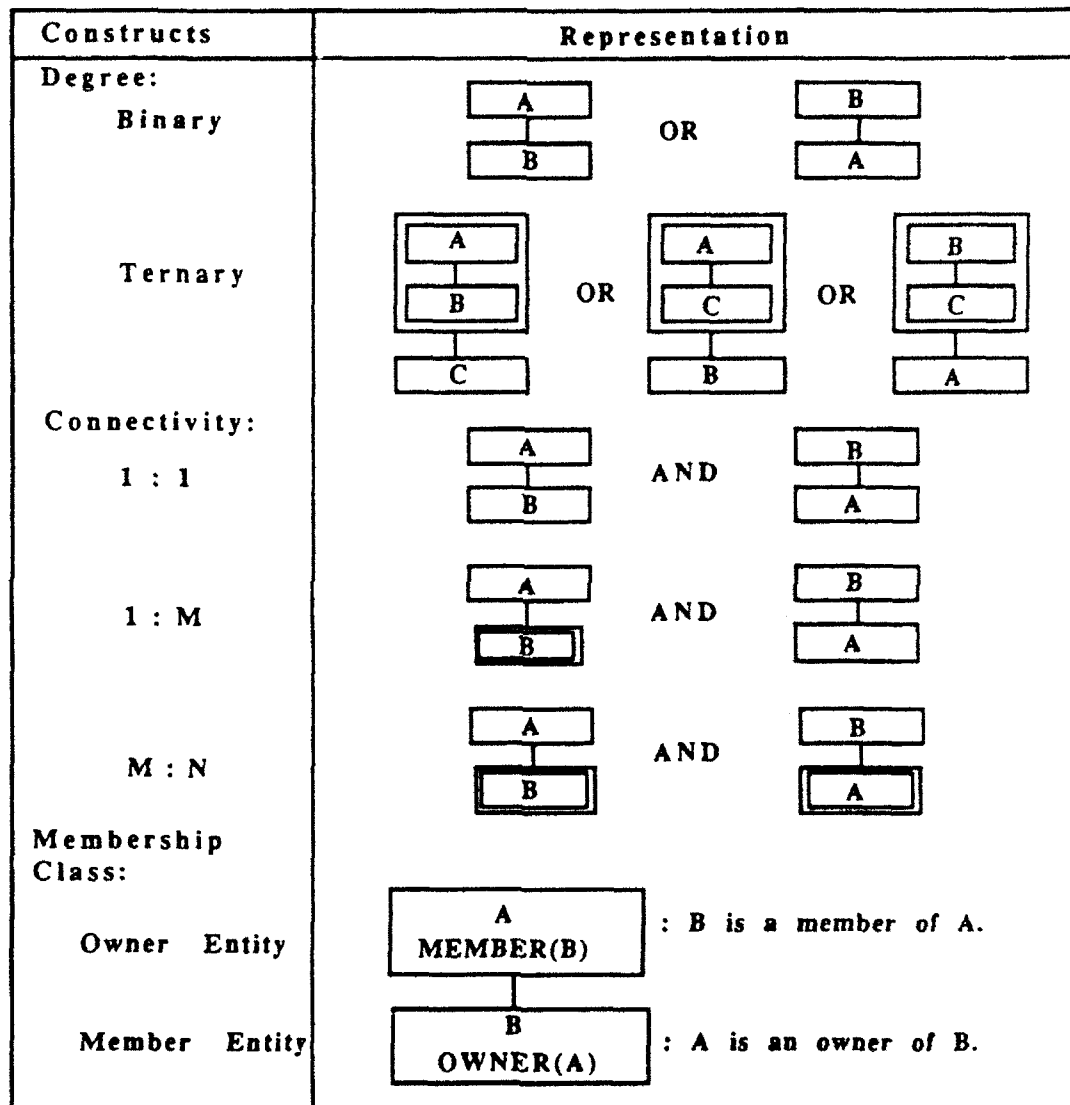


Figure 5. Fundamental Constructs in SOM

2. Connectivity of a relationship specifies the mapping of the associated object occurrences in a relationship. Types of connectivity

for relationships are "one" and "many." The object type (an object or an object set) determines the connectivity of the relationships in SOM. If it is a single object, then the connectivity of relationship is "one," otherwise it is "many."

3. Membership class in a relationship specifies whether either the "one" or the "many" side in a relationship is a member or an owner. If an occurrence of the object with connectivity of one is always required for the occurrence of the other side object, then it is mandatory, otherwise it is optional. The "many" side of a relationship is similarly mandatory if at least one object occurrence must exist, and is optional otherwise. SOM uses constraints methods within an object (frame) to describe the membership type.

4. SOM-based Schema Integration Process.

The SOM-based schema integration process consists of four phases: preintegration, comparison, conflicts resolution, and integration phase. In this section, details of those four phases are discussed.

Preintegration phase: This is a preparation phase for the main integration activities. In this phase all existing schemas are transformed to SOM diagrams. This phase consists of the following two steps.

Step 1. Represent all relations in SOM objects. All existing relations are expected to be normalized. If they are not, they can be normalized using the SOM schema design methodology [Higa and Liu Sheng, 1989]. Each SOM object consists of descriptive attributes and behavioral attributes. All descriptive attributes are copied from a relation, e.g., if relation R1's attributes, including its primary key, are {a1, a2, a3, a4}, then these four attributes are copied into object O1 as shown in Figure 6.

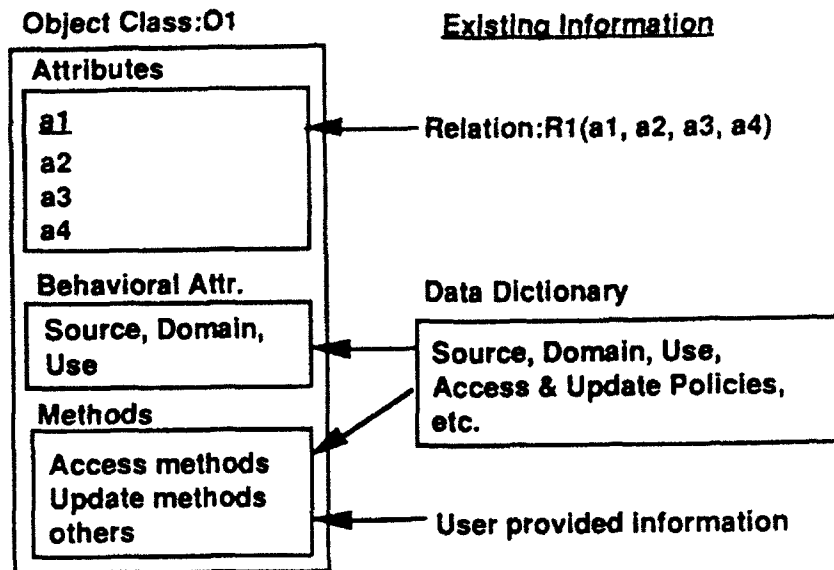


Figure 6. Defining an SOM Object.

Behavioral attributes include object's usage, users, source, synonyms, and domain (see Figure 6.) If data dictionaries or similar documents are available, they will provide those behavioral attributes. If any documents are not available or incomplete then actual users will be able to provide most behavioral attributes. It is worth noting that a complete set of behavioral attributes will help improving accuracy of the schema integration; however, it is not an absolute requirement of this methodology to have a complete set of behavioral attributes.

Step 2. Construct an SOM diagram. SOM objects are put together to form a diagram which represents a schema. The following objects are used in this step to illustrate each situations.

* Primary keys are underlined.

O1={a11, a12, a13, a14};

O3={a11, a41, a31, a32};

O6={a11, a21, a71, a61, a62};

O8={a71, a41, a81, a82, a21};

O10={a71, a102, a103};

O2={a21, a22, a23, a11};

O4={a41, a42, a43};

O7={a71, a72, a73};

O9={a91, a92, a93, a21};

O11={a71, a112, a113};

First, generalization-specialization objects are put together with help from users and existing documents (if no information for those objects is available, there is still a way to find out about them and this technique is discussed at the schema comparison phase.) Then simple objects, which has a single attribute primary key and contains no foreign key, are identified. Simple objects are direct children of a root object in an SOM diagram. Next step is to identify child objects of each simple object. A child object could have a simple relationship (one-to-one and one-to-many,) or a complex relationship (many-to-many and ternary) with its parent. Each relationship can be identified in the following manner:

The one-to-one relationship and the one-to-many relationship are identified when an object contains a primary key of a simple object as its non-prime attribute. For example, O1 is a parent object of O2 since O2 contains O1's primary key but O1 does not contain O2's primary key. Whether they have the one-to-one relationship or the one-to-many relationship must be determined by existing documents or by users.

The many-to-many relationship is identified when an object has a primary key which is a composite of two attributes (i.e., two prime attributes) and one of its prime attributes is a primary key of a simple object. For example, O4 is a parent object of O3 since O3 contains O4's primary key but O4 does not contain O3's primary key. Similarly, O3 belongs to O1 because O3's other prime attribute is O1's primary key. Thus O3 is an intersection object between O1 and O4. The many-to-many relationship exists between O1 and O4. The one-to-many relationship exists between O3 and its parent objects (O1 & O4).

The ternary relationship is identified when an object has a primary key which consists of two or three attributes and at least one of its prime attributes are primary keys of a simple object. If the composite key consists of two attribute, the object must contain one

foreign key. For example, O6 is an intersection object resulted from the ternary relationship among O1, O2, and O7 since its primary key is a composite of three primary keys from O1, O2, and O7. Similarly, O8 is an intersection object resulted from the ternary relationship among O7, O4, and O2.

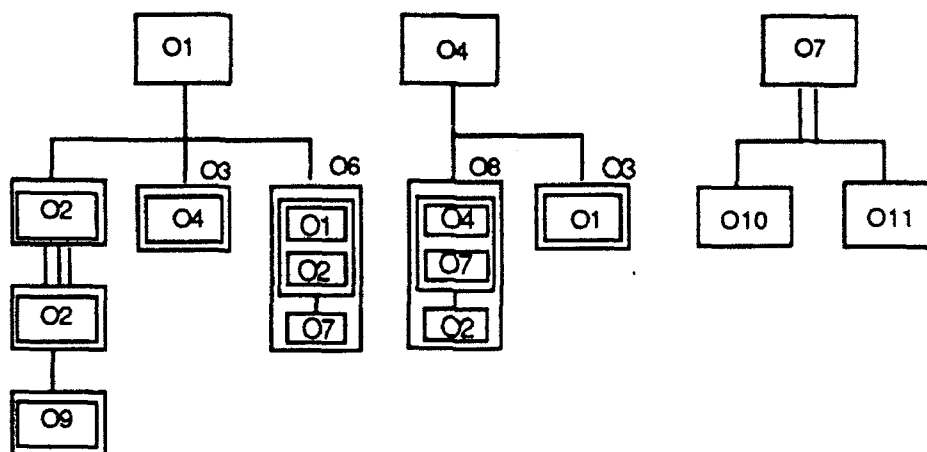


Figure 7. SOM subtrees.

Now we have constructed several SOM subtrees of which each consists of simple objects and their direct child objects (see Figure 7.) For the remaining objects, we can apply the same reasoning to add them to some subtrees. Only one difference is that when two objects have a one-to-one relationship, they contain each others primary key as a non-prime attribute, otherwise their relationship is a one-to-many relationship.

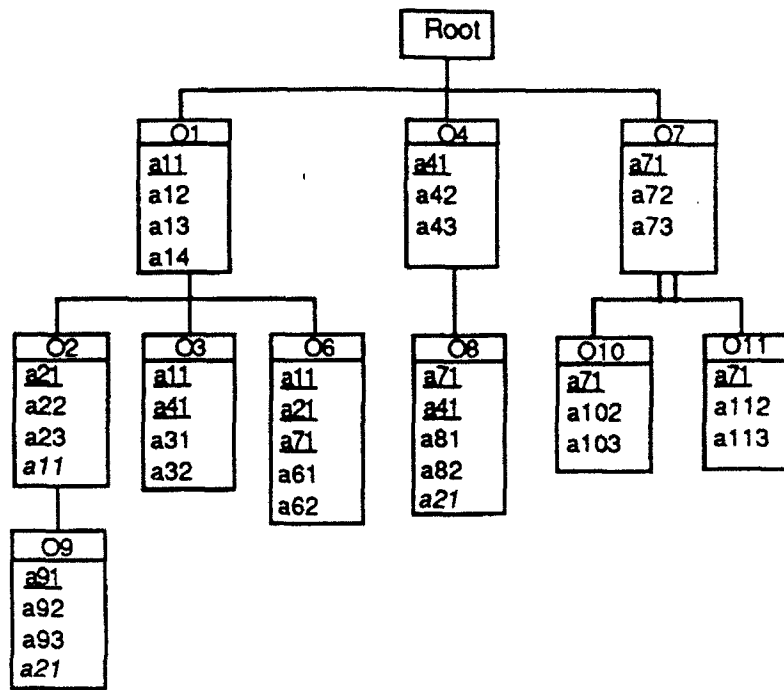


Figure 8. An SOM Representation of A Schema.

After all remaining objects are put to some subtrees, those subtrees are connected by a common dummy object (a root object). The resulted tree represents one DB schema (see Figure 8.) Thus if there are three data bases to be integrated, there are at least three SOM trees to be constructed.

Schema comparison phase: After all schemas are represented in SOM, they are ready for the comparison. Each objects, attributes, structures are tested for similarities, dissimilarities, and conflicts. Since SOM requires each object being defined exactly once at each diagram [Higa and Liu Sheng, 1989], all comparisons discussed in this phase are inter-diagram comparisons. This phase consists of the following two steps.

Step 1. Identification of homonyms.

Candidates of homonyms are easily identified since they share common names. Thus any objects and attributes which have common names are grouped together as potential homonyms. Then

each group of potential homonyms are tested and classified using the following rules.

Definitions for conditional clause:

Common: $A = B$ or A is a subset of B or vice versa;

Similar: A intersects with B ;

Different: A and B are disjoint sets.

Rule 1. IF objects/attributes have a common domain AND
objects/attributes have a common source
THEN objects/attributes are identical.

Rule 2. IF objects/attributes have a common/similar domain AND
objects/attributes have similar/different source
THEN objects/attributes are equivalents.

Rule 3. IF objects/attributes have similar/different domains AND
objects/attributes have a common source
THEN objects/attributes are complements.

Rule 4. IF objects/attributes have different domains AND
objects/attributes have different sources
THEN objects/attributes are homonyms.

Objects/attributes classified by Rule 1 and 2 have strong evidence that they are either identical or equivalents. For such objects/attributes, other attributes such as usage, synonyms, and users will be compared to make the final judgement. Objects/attributes classified by Rule 3 are most likely specializations of some other objects/attributes so they will be recognized as potential specialization groups. Finally, the group classified by Rule 4 is the strong candidates for homonyms. Other behavioral attributes should also be checked for further behavioral dissimilarities.

Step 2. Identification of synonyms. This step consists of two substeps.

Step 2.1. Identification of synonyms by name. If known synonyms of objects/attributes are documented in the existing data dictionary, their names can be compared. Objects/attributes that share common names will be grouped together as potential synonyms. Each group is tested using the following rule.

Rule 5. IF objects/attributes have different names AND
objects/attributes have a common/similar domain AND
objects/attributes have a common/similar source
THEN objects/attributes are synonyms.

Rule 6. IF objects/attributes have different names AND
objects/attributes have different domains AND
objects/attributes have a common/similar source
THEN objects/attributes are specializations.

Step 2.2. Identification of synonyms by structure. For those objects/attributes whose synonyms and some behavioral attributes such as domain and source are not documented to identify additional synonyms, their usage and structures can be compared. The usage of object/attribute describes how it is used, i.e., for computation, for construction of composite data, for determination of status, etc. Standard coding scheme for usage should be established so that the consistent usage comparison is possible. Using usage code and synonyms identified by previous steps, following rule will be applied to identify additional synonyms.

Rule 7. IF objects/attributes have common/similar usage AND
objects/attributes are specializations of a similar
object/attribute OR objects/attributes have
common/similar aspect objects/attributes)
THEN objects/attributes are synonym.

Rule 8. IF objects/attributes are specializations of a similar object/attribute AND objects/attributes have common/similar aspect objects/attributes)
THEN objects/attributes are synonym.

Conflict resolution phase: resolution of both weak and strong conflicts. Conflicts exist among homonyms and synonyms. Different level of abstraction and scale differences which may exist in synonyms are considered to be weak conflicts. Name conflicts of homonyms and data/structure inconsistency of synonyms are considered to be strong conflicts. For the former case, mapping and translation mechanism will be defined to resolve the conflict. An example of a weak conflict resolution is shown below:

O1(a11, a12, a13) and O2(a21, a22, a23) are synonyms where a13 represents temperature in Fahrenheit and a23 represents temperature in Celsius;

Then $a13 \rightarrow a23 = (a13 - 32) * 5/9$ and $a23 \rightarrow a13 = a23 * 9/5 + 32$.

Each translation method will now be attached to corresponding objects shown in Figure 9.

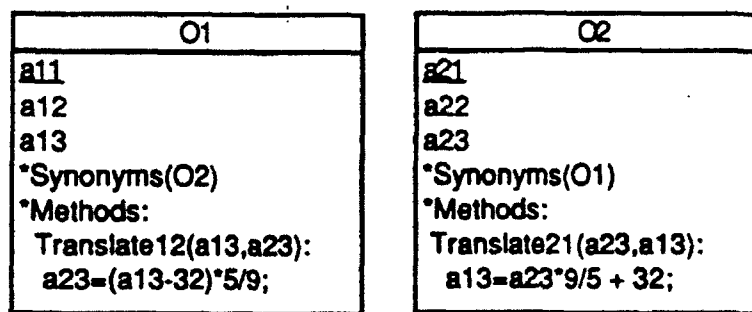


Figure 9. Objects with Translation Method.

For the latter case, neither mapping nor translation is feasible. However, using SOM scheme, name conflicts of homonyms can be

easily resolved. Each object belong to a schema tree in SOM and each SOM tree has unique root object name thus by attaching path object names as prefix, name conflicts will disappear. This point is illustrated in Figure 10.

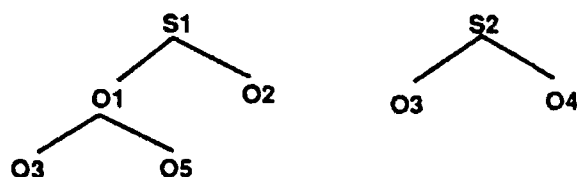


Figure 10. Resolution of Name Conflicts.

In this Figure, O3 in S1 and O3 in S2 are homonyms; however, with their path objects, O3 in S1 is uniquely identified as S1.O1.O3 and O3 in S2 is uniquely identified as S2.O3. Similarly, attribute homonym problems are resolved by attaching its object name as prefix.

Strong conflicts among synonyms are not quite this simple. When they have different data values for the identical object/attribute or different structure for the identical object, one of the following two will be the possible cause for the conflicts:

1. One is correct and others are wrong. In this case, check the source of the object/attribute values and determine the most reliable source. Then distribute the most reliable value to others. Future occurrence of this conflict can be prevented by attaching the value distribution method to the most reliable object.
2. All are correct or others are exceptions of one. In this case, two things are possible:
 - (1) they are specializations -- Identify the common aspects and create a generalization object which contains the common aspects. Then each object keeps its own unique aspect and becomes specialization object of the generalization object.

- (2) one is a general case and others are exceptions -- Identify the object that represents the general case. Then identify the differences between the general case object and other objects. Those differences are the trigger conditions for exceptions. So formulate rules (methods) for exception handling using those conditions and store those conditions at the general case object. Location (or identification) of the general case object should be stored at each exception object.

More detail examples for different types of weak and strong conflicts are discussed detail in Section 5.

Integration phase: A wealth of research results are available in this phase [Batini and Lenzerini, 1984, Navathe et al., 1984, Dayal and Hwang, 1984, Elmasri et al., 1987, Motro, 1987, Desai and Pollock, 1989]. We propose a two-step approach, a mixture of a new approach (integration of SOM diagrams) and an existing approach (verification using functional dependencies.)

Step 1. During the previous phases, all identicals, synonyms, and homonyms are identified and their conflicts are resolved. In this phase, the SOM diagram integration process will be applied to identicals and synonyms. The following discusses the integration process for each case:

Integration of identicals -- In Figure 11, the subtree S1 and S2 are identical thus the schema SA and SB can be integrated based on S1 and S2. The decision of which schema holds the original definition and which other schema copies the original will be most likely an organizational and political issue, i.e., not a technological issue. Therefore, we will not attempt to provide a justification for the decision but will provide a recommendation here. It can be decided based upon two major factors, (1) whichever has the higher stake on the information and (2) whichever provides the minimum cost or maximum benefits (including both tangible and intangible.)

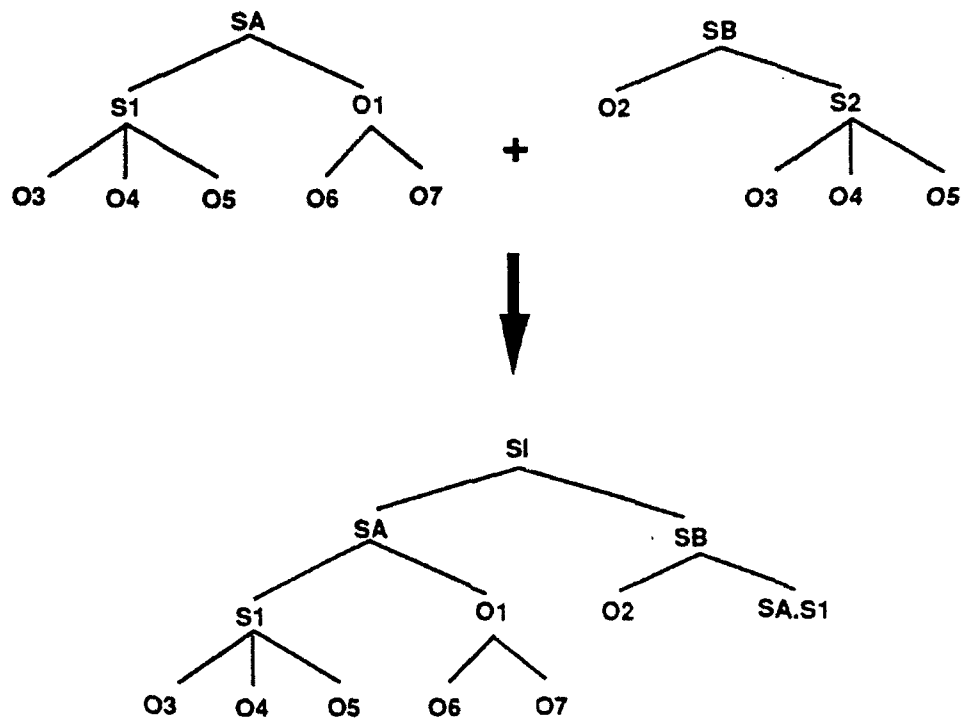


Figure 11. Integration of Identical Objects.

Integration of synonyms I (specializations) -- When an identical specialization exists in multiple schemas, the integration process is similar to that of the integration of identicals. The complete definition of the specialization exists in one schema and others use the stamp coupled definition. However, when multiple version of specialization of an identical object exist, the complete definition of the specialization will be located underneath the root object and other use the stamp coupled definition. For example, in Figure 12, SA.S1 and SB.S1 are identical objects but they have different specialization.

In Figure 12, S1 is directly attached to SI (root) and its two types of specialization, A-spec and B-spec, are attached to SA and SB respectively.

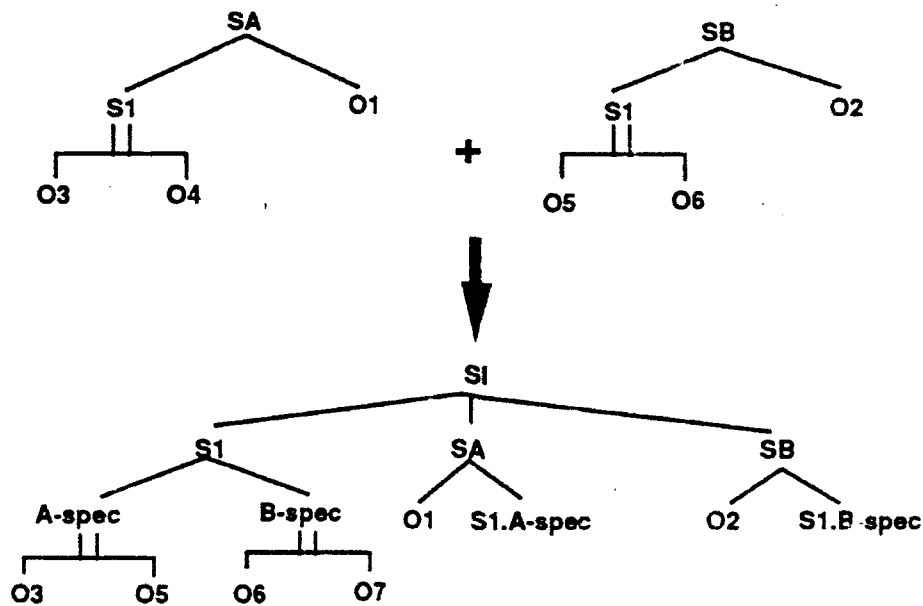


Figure 12. Integration of Specialization Objects.

Integration of synonyms II (exceptions) -- During previous phase, conflict resolution, general case objects and exception case objects are identified, and their corresponding exception handling methods are defined. In order to integrate those objects, exception handling methods must be stored into the general case object. For example, in Figure 13, suppose SA.O1 is the general case object and SB.O1 is the exception case object. SA.O1 can have exactly one O4 and SB.O1 can have multiple O4s. Thus an exception handling method:

"IF parent = SB THEN many O4 Allowed"

will be stored in SA.O1. Then SB uses a stamp coupled O1 object.

Step 2. Verification of the integration using functional dependencies. By checking functional dependencies (FD) among related entities and

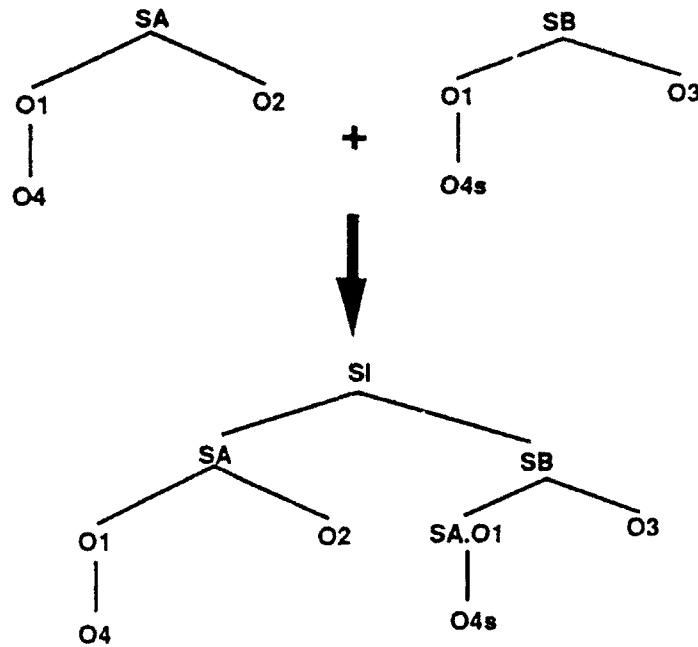


Figure 13. Integration of Exception Objects.

their attributes, the consistency of the integrated schema can be verified. The use of FD for schema integration is logically sound approach and it has been studied by several researchers [Smith et al., 1981 and Dayal and Hwang, 1984,]. However, those research efforts have been mainly focused on algorithmic detection and correction of inconsistent FD (i.e., the problem-solving phase.) Typical FD-based problem-solving method assumes that the problem formulation has already been performed elsewhere. This problem formulation phase of the FD-based integration process is equally difficult (if not more) to the problem-solving phase and is very much neglected research area of the schema integration problem.

In the previous step of this phase, all existing schemas have been graphically integrated. Since all SOM objects have been normalized, FD among attributes can be directly extracted from each SOM object. Also, an SOM diagram is constructed such that all relevant relationships among objects are graphically represented. In addition, transformation/mapping and exception handling methods can be

used to avoid incorrect FD analysis. For example, from Figure 14, following FD among objects can be identified:

for O1 -- O1 → O3 and (O1' or O1''); O1 →> O5;
 for O2 -- O2 → O1'; O2 →> O5 and O4;
 for O3 -- O3 → O5; O3 →> O1 and O4;
 for O1' -- O1' →> O2;
 for O5 -- O5 →> O2 with exception (O5 → O2) when the parent is O3;
 for O4 -- O4 → O3

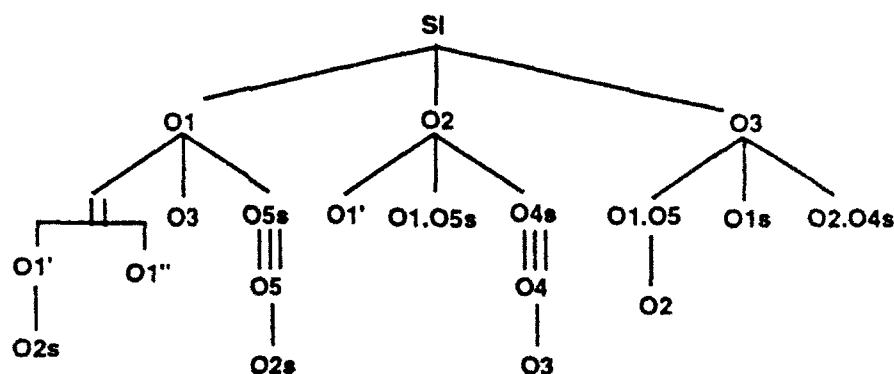


Figure 14. An Integrated SOM Diagram.

Extracted FD are then used by existing FD-based schema integration methods. Therefore, the SOM-based integration methodology supports the problem formulation phase of the FD-based schema integration method.

5. Schema comparison & conflict resolution examples.

In this section, examples from Batini et al. [1986] and a hospital view integration case study [Wu, 1989] are used to illustrate the use of the proposed method in practice.

5.1. Examples from Batini et al.

The article by Batini et al. [1986] is frequently cited in the DB integration field and provides a well defined framework to researchers of this field. In the article, two major types of conflicts (naming and structural) and their subtypes are defined and examples are provided. In this subsection, how the proposed method treats each conflict is described and discussed.

5.1.1. Naming Conflicts.

(1) Homonyms: Figure 15 shows an example of homonyms. The EQUIPMENT in Figure 15a refers to Computers/Copiers/Mimeographic machines, whereas it refers to pieces of furniture in Figure 15b.

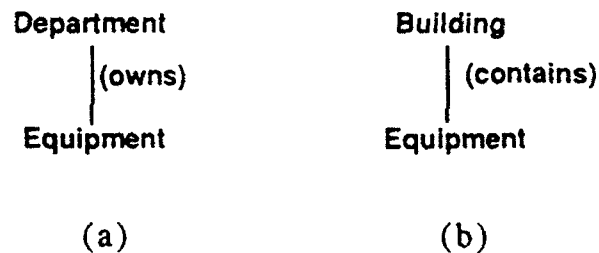


Figure 15. Example of homonyms.

According to Rule 4 in section 4, both objects in Figure 15 are identified as homonyms first and then SOM naming convention (as described in section 4) is applied for each. Thus each object is referred as Department.Equipment for Figure 15a and Building.Equipment for Figure 15b.

(2) Synonyms: Figure 16 shows an example of synonyms. The CLIENT in Figure 16a and the CUSTOMER in Figure 16b both refer to the same object.

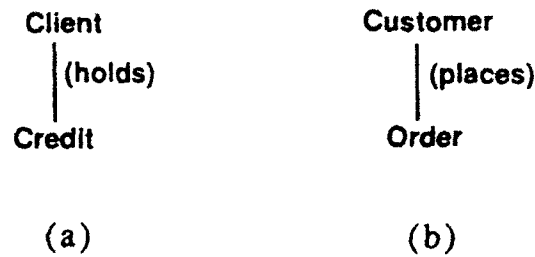


Figure 16. Example of synonyms.

The CLIENT object and the CUSTOMER object have the common domain and source; therefore, according to Rule 5 in section 4, they are identified as synonyms. In SOM, all objects are defined exactly once, the object CLIENT/CUSTOMER is also defined as single object. One name will be used as the primary name and the others are stored in the object as synonyms. The decision of which name to be used as the primary name must be decided by a data administrator.

5.1.2. Structural Conflicts.

(1) Type Conflicts. SEX in Figure 17a is a specialization, where as it is an attribute in Figure 17b.

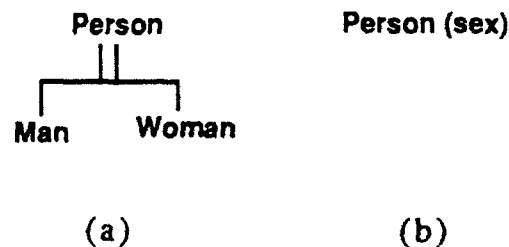


Figure 17. Example of type conflict.

In SOM, any specialization creates a categorization attribute in the parent object, thus the PERSON in Figure 17a also has SEX as an attribute. Therefore, both PERSON objects are identical in SOM.

MAN/WOMAN objects in Figure 17a stores unique features of each objects.

(2) Dependency Conflicts. Batini et al. used marriage between man and woman as an example. It is 1:1 if current marriage is concerned (Figure 18a), but m:n if a marriage history is concerned (Figure 18b).

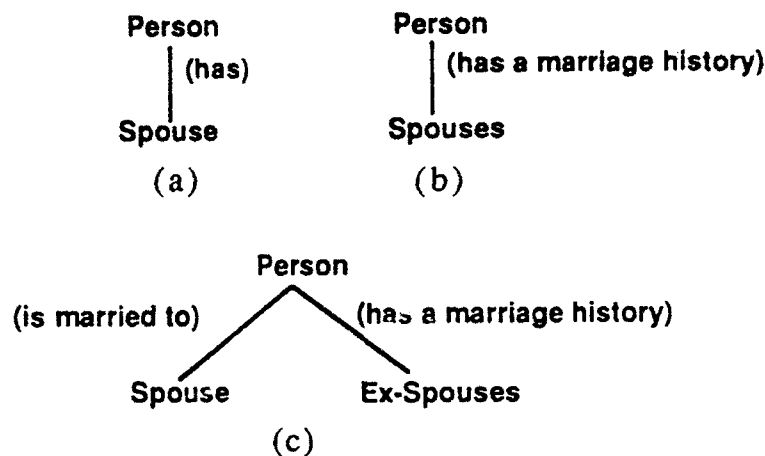


Figure 18. Example of dependency conflict.

In SOM, each aspect relationship represents a unique relationship between objects. When multiple aspect relationships exist between objects, each aspect is represented as a unique association in SOM. As it is clear from Figure 18c, they are different aspects of the object PERSON, thus there is no conflict.

(3) Key Conflicts. Key conflicts occurs when different keys are used for the same object by different applications. In SOM, all objects have exactly one primary key and all other candidate keys are registered as alternate keys (see Figure 19). Both the primary key and alternate keys are treated equally, i.e., the same reference and update constraints will be applied to either types. The most frequently used candidate key will be selected as as the primary key, although the final selection should be made by the database administrator (or by the user group.)

Application A: Employee(emp_id);

Application B: Employee(ssn).

```
Object: Employee{ Attributes[ Primary_key(emp_id),
                               Alternate_key(ssn),
                               Descriptor(name, title, etc.),
                               Foreign_key(proj_id)
                               ], .....
}
```

Figure 19. Example of key conflicts.

(4) Behavioral Conflicts. This type of conflict exists when the same class of objects in distinct schemas have different access and update policies. For example, a department object in one schema (schema A) is allowed to exist without employees, whereas in another schema (schema B), a department object will be deleted when it has no employee. Assuming that either policies are locally correct, there are two possible resolutions in SOM.

(i). Define one policy as the primary policy and the other one as the exception case with an exception handling method. For example, if the schema A is chosen to be the primary policy, then the DEPARTMENT object class will be defined as:

```
Object: Department{
    :
    :
* Methods:
Member(Employee);
Existence_Policy(dept_id, num_of_emp):
    IF num_of_emp = 0 THEN SET(object_id, delete_key, TRUE);
```

```

Access_Policy(schema_id, access_type, . . . ):
  IF schema_id IN (exception_cases) AND delete_key = TRUE
  THEN Deny(schema_id, access_type) WITH Explain("The requested
    object does not exist.");
  }

```

With this definition, a delete mark (`delete_key = TRUE`) is put to the DEPARTMENT object when it receives a message from the EMPLOYEE object that indicates that the last employee of the department is deleted.

In this way, the object is not physically removed, thus it is still accessible through schemas which do not belong to "exception_cases."

(ii). Create specializations if applicable. For example, if two subclass objects, ACTIVE_DEPT and NONACTIVE_DEPT, are created, then the schema A refers to the DEPARTMENT object in Figure 20 and the schema B refers to the ACTIVE_DEPT object in Figure 20.

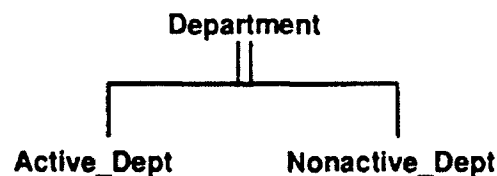


Figure 20. Resolution of behavioral conflicts.

If a behavioral conflict is resolvable by neither of the above resolution, it must be resolved by the database administrator (or by the user group.)

5.2. Hospital View Integration Example.

In this subsection, partial schema integration process from the hospital view integration case study [Wu, 1989] is used to illustrate conflict resolution using SOM. In the case study, there are three

database systems. They are the Hospital Information system (HIS) database for the central management, the Radiology Information System (RIS) database for departmental management, and the Picture Archiving and Communication System (PACS) database for specialized function.

HIS, RIS, and PACS database contains 22, 20, and 16 object classes respectively, and they are partially depicted in Figure 21.

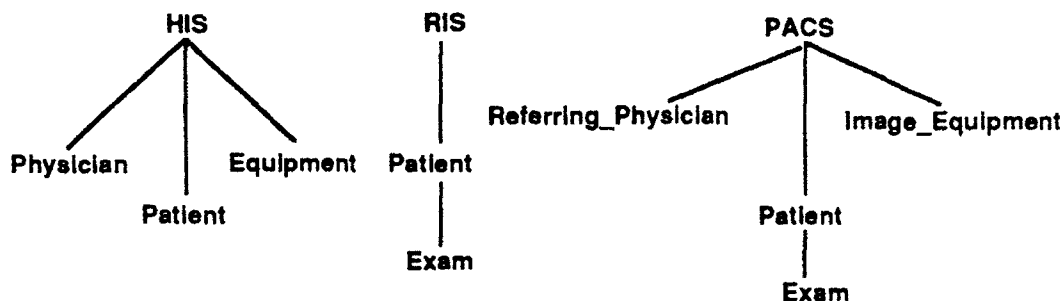


Figure 21. Selected Object Classes for HIS, RIS, and PACS databases.

In Figure 21, HIS.Physician and PACS.Referring-Physician are defined as

```

Object: Physician{ Attributes[ Primary_key(physician_id),
                               Alternate_key(ssn, beeper#, home-phone#),
                               Descriptor(name, dept#, salary, sex, specialty,
                               years-of-experience)];
  
```

```

*Domain(University of Arizona Hospital physicians);
  
```

```

*Source(physicians master file); }
  
```

```

Object: Referring-Physician{ Attributes[ Primary_key(physician_id),
                                           Alternate_key(beeper#, home-phone#),
                                           Descriptor(name, dept#, sex, specialty,
                                           years-of-experience)];
  
```

```

*Domain(University of Arizona Hospital physicians);
  
```

```

*Source(physicians master file); }
  
```

All attributes in PACS.Referring-Physician is identical to HIS.Physician and both have the common domain and source; therefore, PACS.Referring-Physician is a subset of HIS.Physician. Thus an Access_Policy method is added to the definition of HIS.Physician.

Object: Physician{

:
:

*Methods:

Access_Policy(schema_id, access_type):

IF schema_id IN [PACS] THEN MASK(ssn,salary); }

Three Patient object classes are identical thus RIS and PACS refer to HIS.Patient.

An object class "Exam" exists in RIS schema and PACS schema. RIS' Exam is a scheduled exam and PACS' Exam is a performed exam which contains the exam result. Thus they are not identical object class. However, since their path names are unique, RIS.Patient.Exam and PACS.Patient.Exam, they don't create the homonym problem in the integrated schema using SOM.

HIS' Equipment and PACS' Image Equipment shares identical set of attributes as shown below:

Object: Equipment{ Attribute[Primary_key(equipment_id),
Alternate_key(equipment_name),
Descriptor(status, description)];

*Domain(central medical equipments);

*Source(central medical equipment resource file) }

Object: Image Equipment{ Attribute[Primary_key(equipment_id),
Alternate_key(equipment_name),
Descriptor(status, description)];

*Domain(radiology departmental image equipments);
*Source(radiology department medical equipment resource file) }

Thus they are candidate for synonyms. However, they have different domains and sources, hence they are not considered as synonyms. Interviews with users on these object classes revealed that "Image Equipment" is a subset of "Equipment." However, all image equipments in PACS are strictly departmental equipments and are managed solely by the radiology department. Thus they are kept as separate object classes in the integrated schema.

The resulting integrated schema is depicted in Figure 22.

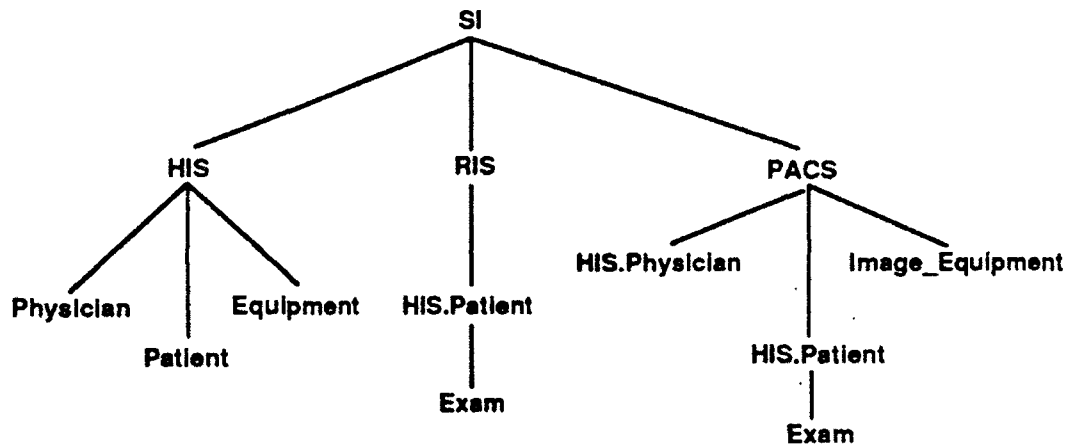


Figure 22. The Integrated Hospital Schema.

References

- [1] Batini, C. and Lenzerini, M., "A Methodology for Data Schema Integration in The Entity Relationship Model," *IEEE Transactions on Software Engineering*, 10(6), November 1984, pp. 650-663.
- [2] Batini, C., Lenzerini, M., and Navathe, S.B., "A Comparative Analysis of Methodologies for Database Schema Integration," *ACM Computing Surveys*, 18(4), December 1986, pp. 323-364.
- [3] Belogus, D., Zeigler, B.P., and Bolsoi, A., "ESP- an interactive tool for system structuring," In *Proc. of the European Meeting on Cybernetics and System Research*, Hemisphere Press, 1980.
- [4] Chee, K.H., "SemTool: An Interactive Graphical Database Modeling Software Using The Structured Entity Model Approach," Master's degree project, the University of Arizona, MIS Department, March 1989.
- [5] Dayal, U. and Hwang, H.Y., "View Definition and Generalization for Database Integration in a Multidatabase System," *IEEE Transactions on Software Engineering*, 10(6), November 1984, pp. 628-644.
- [6] Desai, B.C. and Pollock, R., "On Schema Integration in a Heterogeneous Distributed Database Management System," In *Proc. of Compsac '89*, September 1989, Orlando FL., pp. 218-224.
- [7] ElMasri, R., Larson, J., and Navathe, S.B., "Integration Algorithms for Federated Databases and Logical Database Design," *Tech. Rep.*, Honeywell Corporate Research Center, 1987.
- [8] Higa, K., "Object-based Requirements Analysis for End-User Database Development," working paper, Georgia Institute of Technology, College of Management, March 1990.
- [9] Higa, K. and Liu Sheng, O.R., "An Object-Oriented Methodology for End-User Logical Database Design: The Structured Entity Model Approach," In *Proc. of IEEE Compsac '89*, Orlando FL., September 1989.
- [10] Motro, A., "Superviews: Virtual Integration of Multiple Databases," *IEEE Transactions on Software Engineering*, 13(7), July 1987, pp. 785-798.

- [11] Navathe, S.B., Sashidhar, Elmasri, R., "Relationship Merging in Schema Integration," In Proc. of Conference on VLDB, Singapore, August 1984, pp. 78-90.
- [12] Schneyer, R., Modern Structured Programming, Mitchell Publishing, Inc., Santa Cruz, CA. 1984, pp. 121-122.
- [13] Smith, J.M., Bernstein, P.A., Dayal, U., Goodman, N., Landers, T., Lin, K.W.T., and Wong, E., "Multibase-integrating heterogeneous distributed database systems," In Proc. of the National Computer Conference, Vol. 50, 1981, pp. 335-347.
- [14] Teorey, T.J., Yang, D., and Fry, J.P., "A Logical Design Methodology for Relational Databases Using The Extended Entity-Relationship Model," Computing Surveys, 18(2), June 1986.
- [15] Wu, H.Y., "View Integration on a Hospital Case," case study report, the University of Arizona, MIS Department, December 1989.